EV355229096

Software Updating System and Method

Inventors:

Robert T. Wickham,

Vinay Deo,

Shafqat U. Khan,

Shantanu Sardesai,

and

Adam Welker

ATTORNEY'S DOCKET NO. MS1-1552US

SOFTWARE UPDATING SYSTEM AND METHOD

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/455,197, filed March 17, 2003, entitled "Software Updating System and Method", which is hereby incorporated by reference. This application also claims the benefit of U.S. Application No. 10/385,391, filed March 10, 2003 and still pending, entitled "Software Updating System and Method", which is hereby incorporated by reference.

TECHNICAL FIELD

[0001] The disclosure relates to installation of software updates.

BACKGROUND

[0002] A very large number of software applications, programs and files used by computers need periodic updates, which are frequently referred to a "patches". Many updates are of vital importance, such as those having to do with security or application functionality.

[0003] Installing and maintaining software within an enterprise network environment is a major cost to corporations and organizations. This is particularly the case where large numbers of networked computers are involved, each computer having a large number of programs and each program having a large number of possible revision levels. As the number of computers within a network increases, and the number of files, programs and associated versions of the files and programs on each computer also increases, it becomes

progressively harder to maintain each computer in a condition that maximizes security and functionality.

[0004] Additionally, workstation down time is increasingly an issue, due in part to the time required to reboot computers after application of an update. The known process by which client computers are updated has created time consuming cycles during which updates are applied and the computer is rebooted. Accordingly, productive time is lost.

[0005] Moreover, the IT technician is not provided with any metrics describing important data related to the time that updates become available, the time such updates become approved, and the time taken to fix such hazards by installing an update. Consequently, the progress toward the goal of timely application of all updates is not well understood or aggressively pursued at present.

SUMMARY

[0006] In one embodiment, software updates are applied to one or more client computers. The one or more client computers are assigned a level of service governing aspects of the application of the updates. The scheduling of the application of the software updates is done according to the level of service. The application of the software update is then performed according to the schedule.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The following detailed description refers to the accompanying figures. In the figures, the left-most digits(s) of a reference number identifies the figure (Fig.) in which the reference number first appears. Moreover, the same reference numbers are used throughout the drawings to reference like features and components.

[0008] Fig. 1 is a block diagram describing the configuration of an exemplary software updating system.

[0009] Fig. 2 is a diagram that describes the structure and operation of a vulnerability matrix.

[0010] Fig. 3 is a flow diagram that describes the operation of an exemplary software updating system.

[0011] Fig. 4 is a flow diagram that describes the operation of an exemplary catalog synchronization module on a server.

[0012] Fig. 5 is a flow diagram that describes the operation of an exemplary scan tool on a client computer.

[0013] Fig. 6 is a flow diagram that describes the operation of an exemplary scan engine on a client computer.

[0014] Fig. 7 is a flow diagram that describes the operation of an exemplary software update approval tool on a server.

[0015] Fig. 8 is a flow diagram that describes the operation of an exemplary software update installation tool on a client computer.

[0016] Fig. 9 is a flow diagram that describes the operation of exemplary grace periods and enforcement periods on a client computer.

[0017] Fig. 10 is a flow diagram that describes exemplary conditions controlling rebooting a client computer.

[0018] Fig. 11 is a flow diagram that describes exemplary mapping of success codes resulting from update installations on a client computer.

[0019] Fig. 12 is a flow diagram describing an exemplary method by which the user may be given some control over the timing of the process by which updates are installed on the user's computer.

[0020] Figs. 13 through 16 are exemplary user interfaces associated with the method of Fig. 12.

[0021] Fig. 17 is a flow diagram describing an exemplary method by which updates may be applied to a large number of systems in an orderly manner.

[0022] Fig. 18 is a flow diagram describing an exemplary method by which the efficiency of update installation may be improved.

[0023] Fig. 19 is a flow diagram describing an exemplary method by which un-trusted updates may be tested and evaluated in an efficient manner.

[0024] Fig. 20 shows an update package partitioned to divide trusted and un-trusted updates.

[0025] Fig. 21 is a flow diagram describing an exemplary method by which a template may be constructed from a reference computer for use in updating computers having a software configuration based on the reference computer.

DETAILED DESCRIPTION

[0026] Fig. 1 shows a network environment within which an exemplary software updating system 100 may operate. A server 102 is connected to a client 104 by any type of network 106, such as the Internet or an intranet within an enterprise. The server 102 is typically configured with an operating system 108. Software configured for asset inventory, software distribution and infrastructure security, such as Microsoft Systems Management Server (SMS) site server 110 or similar product may also be present. Additionally, an XML parser 112 is installed.

[0027] A catalog synchronization module 114 allows the server 102 to obtain an update catalog 116 from an update authority 118. The update catalog 116 may be configured as an XML document, and includes information about the availability of software updates ("patches") and the version of the software to which they should be applied. Additionally, the update catalog 116 may include complex rules, typically in the form of Boolean logic, which prescribes the conditions under which individual software updates should be installed. In an over-simplified example, the rules may indicate that an update should be installed if the target application is greater than revision 2, but only if revision 4 of another program is present. The update authority 118 may be a trusted software source, such as Microsoft Corporation, wherein the trusted software source maintains information concerning software updates.

[0028] Therefore, the catalog synchronization module 114 is configured to communicate with the update authority 118, to maintain the resident copy of the update catalog 116 in current form. Additionally, the catalog synchronization module 114 is configured to check for a code (an "authenticode") to determine if the update catalog 116 is authentic, or has been corrupted, tampered with or otherwise rendered useless or harmful.

[0029] An exemplary software update approval tool 120 may be configured as a wizard to guide a user (such as an IT (information technology) technician) on the server 102 through the deployment of software updates on one or more client computers 104. In particular, the software update approval tool 120 is configured to allow the user to approve or reject application of software updates for installation on one or more client computers 104, wherein the updates were recommended by audit data 122 received from the client 104. Information disclosing the file configuration on the client—including file versions present and the present update level—is obtained from audit data 122 which was transferred from the client to the server via SMS, as will be seen in greater detail below.

[0030] In one embodiment of the software update approval tool 120, the approval process may involve presenting the user with a user interface 200, such as that seen in Fig. 2. By viewing the user interface, the IT technician may understand in detail the vulnerabilities facing each client computer 104 on an enterprise network. In an optional configuration, a vulnerability matrix 202 showing the update status for a client computer 104 is present. The matrix 202 may be configured by arraying representations 204A—204N of the files present on the particular client on one axis, and representations 206A—206M of the updates (patches) associated with the files 204 on another axis. The matrix 202 includes representations 208 indicating that a given update was applied to a given file; representations 210 indicating that the update was not applied to the file; and representations 212 indicating that the update is not applicable to the file. By viewing the vulnerability matrix 202, the user may appreciate the vulnerabilities associated with a file present on the client computer 104, as well as whether the vulnerabilities have been eliminated or mitigated by application of the update(s) associated with the file.

[0031] Returning to Fig. 1, the software update approval tool 120 allows the IT technician on the server 102 to approve updates for application to the client 104. The software update approval tool 120 is configured to, upon approval of an update, locate and obtain a copy of required updates 124. Accordingly, such approved updates 124 may be obtained by the approval tool 120 from a download center 126. Information on the location of the download center, such as a URL (uniform resource locator) may be obtained from the update catalog 116. Updates obtained for a client may then be replicated to the client using SMS 110 or other software distribution technology.

[0032] An exemplary client 104 is configured with an operating system 128 and XML parser 130. SMS client software 132 or similar asset inventory and software distribution software and WMI 134 (Windows® management instrumentation) software 134 or similar are also installed.

[0033] A copy of the update catalog 136, received from the server 102 via SMS, provides information regarding the relationship between files potentially present on the client and updates which may need to be installed on those files. A scan tool 138 is configured to call a scan engine 140. Where the scan engine 140 is a program, the scan tool 138 invokes the program. Where the scan engine has been advantageously configured as an API (application programming interface) the scan tool 138 calls the API.

[0034] The scan engine 140 is configured to read the update catalog 136, which is typically in the form of an XML document, using the XML parser 130. The scan engine 140 is then configured to query the operating system 128, to determine the exact revision level of programs and/or files present on the client 104. The scan engine 138 may base each query in part on the outcome of previous queries and on the rules—typically expressed as Boolean equations—within the update catalog 136. Accordingly, the scan engine 140 is

configured to determine the file type and revision level of all relevant files on the client 104, and additionally to determine the updates that are applicable to the files found. When completed, this information becomes audit data 142.

[0035] The scan tool 138 is configured to save the audit data 142 into a queryable interface with the server 102. WMI 134 is such an interface, and a preferred scan tool 138 is configured to save the audit data 142 into WMI 134 and to provide an indication to the SMS client 132 that will cause the audit data 142 to be replicated to the server 102, where it may be aggregated with the audit data of other clients at 122.

[0036] A scan tool helper 144 may be configured separately or as part of the scan tool 138. The scan helper 144 manages the process of merging the results of the audit data 142 with service level data. Such service level data may be reflected in modifications to the Win32_PatchState schema or similar data structure. In an exemplary environment, modification to the Win32_PatchState schema may include the addition of fields including: string AuthorizedName (name of the update), datetime TimeDetected; datetime TimeApplied, and datetime TimeAuthorized. Field TimeDetected records the time at which an available update is discovered, thereby implying a time at which a potential security breach is detected. Field TimeAuthorized records the time at which the server 102 (i.e. the IT department administration) authorized the application of the update. Accordingly, the time required by the IT department to act can be derived by comparing the TimeDetected from the TimeAuthorized. Field TimeApplied records the time at which the update was applied, and is an overall measure of the responsiveness of the IT department, and is also a measure of the IT department's responsiveness in installing the update after authorizing the update. Using these modifications to the Win32_PatchState schema, the scan tool helper 144 is able to generate the

above service level data associated with TimeDetected, TimeAuthorized and TimeApplied.

[0037] A software update installation agent 146 is configured to install the approved update(s) 150 sent by the server 102 via SMS or similar distribution application. In a preferred embodiment, the software update installation agent 146 is configured to call the scan engine 140. The scan engine 140 is configured to again evaluate the files on the client 104 with respect to the update catalog 136, which contains information on updates available, files needing updates, and rules for determining the which updates apply to which files. The output of this second scan—current audit data—of the client's files is put into the scan tool cache 152. The software update installation agent 146 is configured to use the scan tool cache 152 to prevent the installation of updates that are no longer warranted. The installation of updates may become unwarranted, such as in the circumstance where changes in the client computer's file system have occurred more recently then the transmission of the audit data 142 to the server 102. Examples of such a change include installing an updated version of a program, thereby replacing the earlier version which needed an update.

[0038] A preferred software installation agent 146 is configured to review the exit codes generated by the installation of the updates 150. A success code table 148 is configured to map a wide variety of initial exit codes into an output exit code that more correctly reflects the underlying meaning of the initial exit code. While zero is traditionally used to indicate a successful update install, and other numbers enumerate different potential errors, this is nomenclature is not always accurate. In some cases, non-zero exit codes indicate a satisfactory update installation due to situations unforeseen by the author of the update or the update installation technology. Such situations

include factors related to the software configuration on a particular client. Accordingly, the success code table 148 is configured to map the exit codes generated by the installation of updates into exit codes which more accurately reflect the situation which caused generation of the exit code and provide consistency in success code or exit code meaning.

[0039] In particular, differences between exit codes that were generated by different install engine technology may exist. For example, Windows® Installer, Update.EXE and I-Express may not be in complete agreement on the precise meaning of a given exit code. Accordingly, the success code table 148 may be configured to map success codes in any desired manner, such as to map numerically different codes having the same meaning into a consistent numeral, thereby resulting in consistent success code interpretation. Subsequent to exit code consideration, those items resulting in immediate availability will be re-scanned to confirm their availability. Items which result in a computer reboot shall have their availability confirmed by re-scanning immediately following the reboot.

[0040] In some cases, identification of an update's association with a particular install engine technology provides sufficient information to map the success codes associated with the update. In other circumstances, the success code table 148 must be edited by an IT professional to reflect a special case associated with an particular update. Accordingly, the success code table 148 is editable, configurable and extensible, and can be modified to allow mapping of the success codes generated by any update's installation process into a consistent meaning.

[0041] Fig. 3 is a flow diagram that describes an exemplary method 300 by which the software updating system 100 may be operated. At block 302, an update catalog 116 is maintained on the server 102 by a catalog

synchronization module 114 or similar. As a part of the maintenance of the update catalog 116, the catalog synchronization module 114 downloads a fresh copy of the update catalog at regular intervals. Each time the update catalog 116 is downloaded, the synchronization module 114 checks applicable codes to determine if the catalog downloaded is authentic. All or any relevant part of the update catalog 116 may be sent at regular intervals to the client 104 for storage at 136.

[0042] At block 304, audit data 142 is generated on the client 104 based on queries made to the operating system 128 and based on information from the update catalog 136. The audit data 142 may be generated under the direction of a scan tool 138 by operation of a scan engine 140. The audit data 142 can include an inventory of files within the client computer's file system which need updates and an indication of the applicable update for each file.

[0043] At block 306, the audit data 142 may be aggregated with the audit data of other clients on the server 102 in an audit data library 122. The audit data is analyzed, such as by a software update approval tool 120, to determine if each update should be installed. The approved updates are then sent to the appropriate client by SMS or other means.

[0044] At block 308, the approved updates 150 are installed on the client 104. The installation process may be performed by a software update installation agent 146 or similar procedure. The scan tool cache 152 is consulted to prevent the installation of updates for which there is no longer a need. Changes in the need for an update could be related to the recent installation of a newer version of a program, for example. The client may be rebooted after installation of one or more updates. The decision to reboot may be governed by dynamic restart detection, as seen in greater detail in Fig. 10. Success codes resulting from the update installation may be mapped according

to the success code table 148, as seen in greater detail in Fig. 11. Success code information is utilized in the creation of service level data, which can be used, for example, to provide feedback to a corporate IT department. Such service level data can include time measurements between availability of an update and authorization of the update by the IT department and between availability and installation of the update.

[0045] Fig. 4 is a flow diagram that describes an exemplary method 400 by which the catalog synchronization module 114 may be operated, thereby more fully explaining the operation of block 302 of Fig. 3.

[0046] At block 402, the server 102 periodically downloads an update catalog 116 from an update authority. The download may be managed by the catalog synchronization module 114 or similar structure, thereby maintaining the copy of the update catalog 116 in a current condition. The update catalog 116 is typically in the form of an XML document, and contains information about available updates, the files and file versions to which the updates apply, and rule governing such application. Due to the complexity of the rules, they may be expressed in the form of Boolean equations.

[0047] At block 404, in a typical download of the update catalog 116, the catalog synchronization module 114 checks available codes associated with the update catalog 116 as a conformation of the catalog's authenticity.

[0048] At block 406, the update catalog 116 is stored as an SMS package. SMS is told that the package includes changes, i.e. that the package is new. Accordingly, SMS replicates the update catalog 116 to all clients 104.

[0049] Fig. 5 is a flow diagram that describes an exemplary method 500 by which the scan tool module 138 on the client 104 may be operated, thereby more fully explaining the operation of block 304 of Fig. 3.

[0050] At block 502, the scan tool 138 calls the scan engine 140. In an exemplary configuration, the scan engine 140 is a program that must be called. However, the scan engine 140 may advantageously be configured as an API, thereby simplifying the operation of the scan tool 138.

[0051] At block 504, the scan engine 140 uses the update catalog 136, typically in XML form, to perform the audit of the software on the client 104. This audit, described more fully with reference to Fig. 6, results in the production of audit data 122.

[0052] At block 506, the scan tool 138 saves the audit results 142 received from the scan engine 140 into a queryable interface with the server 102. In a typical application, the queryable interface with the server is WMI 134 (Windows® management instrumentation). Accordingly, the scan tool 138 populates the audit results into a WMI repository 134 on the client 104. The audit results 142 are then uploaded by SMS to the server 102, typically for storage in a library 122 with the audit results of other clients.

[0053] Fig. 6 is a flow diagram that describes an exemplary method 600 by which the scan engine 140 on the client 104 may be operated, thereby more fully explaining the operation of block 504 of Fig. 5.

[0054] At block 602, the scan engine 140 reads the update document 150, which typically contains an XML document describing files, updates and associated rules of application.

[0055] At block 604, the scan engine 140 queries the operating system 128 of the client 104 to determine the existence of files to which reference was made in the update document 150. At block 606, the rules contained within the update document are then applied, thereby determining the files to which an update applies and the identity of that update.

[0056] At block 608, the scan engine 140 assembles the identities of the files needing an update and the associated updates, and returns this information to the scan tool 138 as the audit data.

[0057] Fig. 7 is a flow diagram that describes an exemplary method 700 by which a software update approval tool 120 may be operated on the server 102, thereby more fully explaining the operation of block 306 of Fig. 3. The exemplary method 700 allows for the selection of approved software updates, which are then replicated to the appropriate client computers for application.

[0058] At block 702, a user interface is opened, thereby allowing an IT technician using the server 102 to view audit results for one or more client systems. At block 704, optionally, a vulnerability matrix 200 is generated and displayed. The vulnerability matrix 202 shows client file inventory, recommended updates and indicates if the update has been applied. At block 706, the user interface allows the IT technician to indicate the approved updates (patches). The approval process could be presented to the user in the form of a wizard, or other applicable format. The approval could be based in part on the vulnerability matrix 200 the technician viewed. At block 708, in some applications, the user interface is restricted to allow selection of updates only from an approved group of updates having passed testing on a test collection of computers. If application of the update to the test collection of computers was successful, then the updates are included among those from which the IT professional may select for installation on a given system. In some applications, the IT professional may be challenged by the user interface to state the name of a test collection of systems upon which the update was installed in the testing process, or other details which confirm that testing was performed.

[0059] At block 710, the approved updates 124 are obtained, perhaps from an update download center 126. At block 712, suppression of reboot may be indicated for some updates. At block 714, dynamic rebooting may be indicated for other updates. Fig. 10 discusses rebooting in greater detail. At block 716, the updates are then transferred to the appropriate client 104 via SMS or other file transfer utility.

[0060] Fig. 8 is a flow diagram that describes an exemplary method 800 by which a software installation agent, such as agent 146, may be operated on the client 104, thereby more fully explaining the operation of block 308 of Fig. 3. The exemplary method 800 installs the updates sent by the server 102 onto the client 104.

[0061] At block 802, a user interface may be opened on the client 104 to announce to the user the opportunity to have updates installed at this time. At block 804, where a user interface has opened, a countdown timer causes appropriate default actions to be taken upon expiration of a timed period if no one is present at the client computer. The default action may be set by the corporate IT department, and may include "install" and "postpone". Where the default is "install," the installation of the updates proceeds after the timed period has expired. Where the default action is "postpone," the installation is delayed. At block 806, where a grace period has ended and the enforcement period has started, the "postpone" option is eliminated, and may appear to be a "grayed" option on the user interface. In this circumstance, the remaining default, "install," is invoked, and the installation of the updates proceeds. The identifying name of the corporate IT department is affixed to this user interface to ensure the employee is aware of whom the computer's maintenance has been entrusted.

[0062] At block 808, the appropriate scan tool 138, 140 is executed to refresh the scan tool cache 152. By refreshing the scan tool cache 152, it can be determined if the status of files on the client 104 has changed, thereby changing the need for one or more updates to be installed. The audit results of the scan tool cache 152 may be deposited into a queryable interface, such as WMI.

[0063] At block 810, the WMI class is queried, and the results (i.e. the updates and their associated target files) are intersected with the list of approved updates received from the server. Accordingly, where WMI indicates that there is no longer a need to install a given update, or where the server did not forward an update (perhaps due to approval rejection by the IT technician) one or more updates will not be installed.

[0064] At block 812, each update to be installed is checked to determine if an enforcement period has been reached. In some cases, the IT department operating the server 102 will provide a grace period, followed by an enforcement period. During the grace period, the update can be rejected by the client. During the enforcement period, the update cannot be rejected by the client.

[0065] At block 814, for each approved update that is enforced and still applicable (in view of the scan cache), the update is installed using the meta-data included in the XML file (the update document 150).

[0066] At block 816, following install, a status message is issued for each update, and a summary message reflecting the overall status of the evaluation/installation cycle is issued. In one embodiment, the status message will observe a SuccessCodes mapping table feature, wherein mapping of non-zero exit codes into success states is allowed. Success code mapping is discussed in greater detail in the discussion of Fig. 11. At block 818, the

success codes may be included in service level data, which may be transmitted to the software update approval tool 120 or other location on the server 102.

[0067] At block 820, the WMI class is updated to reflect newly installed updates, and the SMS inventory process is conditionally started. Additionally, service level data may be transmitted to the software update approval tool 120 or other location. For example, service level data may include information indicating the length of a time period between update availability and update installation.

[0068] At block 822, determination is made if a system restart is needed, and a check is made to ensure that the system is permitted to be restarted. Additional detail associated with system restarting is seen in the discussion of Fig. 10.

[0069] Fig. 9 is a flow diagram that describes an exemplary method 900 by which grace periods and enforcement periods may be employed to balance the need to install updates with the need for knowledge workers to efficiently utilize their computers.

[0070] At block 902, the IT department associates a grace period with an update, after which an enforcement period is scheduled. At block 904, the update installation process is started for an approved update having assigned grace and enforcement periods. At block 906, a computer for which an update is scheduled is found on a network—or returned to a network, such as in the case of a laptop computer)— thereby triggering the start of the grace period. At block 908, the user may elect to invoke the grace period to delay installation of the update. This action allows the user to continue working. At block 910, optionally, the user may permit installation of one or more updates, which may be followed by a reboot. At block 912, where the grace period ends prior to permitting the installation of one or more updates, the enforcement period

begins. The enforcement period forces the user to accept the installation of the update. At block 914, the update(s) is/are installed, possibly followed by a reboot.

[0071] Fig. 10 is a flow diagram that describes an exemplary method 1000 by which dynamic restart detection may be performed, thereby increasing the efficiency of the update process. At block 1002, updates may be associated with rebooting instructions, thereby informing the software update installation agent 146 of the rebooting requirements. At block 1004, where an update was designated as an exclusive primary update, the update is applied individually prior to other updates, to be followed by a reboot. Examples of such updates include service packs, which combine many updates. Application of such an update may obviate the need for other updates; accordingly, such an update should be applied first. At block 1006, updates designated for exclusive installation and reboot are installed individually and the system is rebooted. An example of such an update is a video driver, which is sufficiently important that it is preferably installed individually prior to a reboot. At block 1008 where an update was designated for automatic reboot detection, a reboot is performed only if indicated by conditions found after the install. Such conditions may include the discovery of files which are left over by updated applications, and which indicate the need for a reboot. At block 1010, where a reboot was designated as being "always" or "never" indicated, the reboot is performed or not performed, as indicated.

[0072] At block 1012, where suppression of the reboot was indicated, the reboot is delayed until an appropriate time. Where suppression is indicated, if is frequently advisable to time the application of the updates a short time before a scheduled reboot. This is particularly true where the install is being

performed on a server, which may have a rigorous schedule governing system rebooting which is intended to maximize uptime.

[0073] Fig. 11 is a flow diagram that describes an exemplary method 1100 by which a success code table 148 may be maintained, and by which success code mapping may be performed in a manner which maps success codes having misleading or inconsistent meaning to success codes having an expected and/or appropriate meaning.

[0074] At block 1102, following installation of an update, a success code table is consulted, and the success code resulting from the update installation is checked.

[0075] At block 1104, where the update installation was part of a testing procedure, the IT technician may edit the success code table 148. At block 1106, the success code table 148 may be organized by groups, where the updates in each group have success codes with similar meaning. For example, each groups of updates may be configured for installation by similar update installation engine technology. At block 1108, the IT technician may further edit the success code table to accommodate the installation of updates resulting in codes which are exceptions to general rules.

[0076] At block 1110, where the installation of the update was not associated with a testing procedure, location of the success code resulting from the installation within the success code table 148 allows mapping of the success code. Accordingly, the result of the mapping provides information that is consistent with expectations.

[0077] Fig. 12 is a flow diagram describing an exemplary method 1200 by which the user may be given some control over the timing of the process by which updates are installed on the user's computer. More particularly, the method 1200 may allow the user to control the timing of the operation of the

scan tool module 138, the software update installation agent 146 and/or the rebooting of the user's computer following the installation of the updates 150.

[0078] At block 1202, a notification icon is presented to the user at all times for the purpose of allowing the user to gain information about, and control over, the timing and nature of updates to be performed to the user's computer. The icon can be configured to appear similar to a speaker icon, typically found in the lower right of current Windows® operating systems. At block 1204, in an exemplary user interface, right clicking the icon provides the user with an interface 1300 providing a choice of "Display Reminder" and "Install Software Updates" as seen in Fig. 13. At block 1206, selection of the "Display Reminders" button seen in Fig. 13 may result in the exemplary "Reminder" interface 1400 seen in Fig. 14. The interface 1400 of Fig. 14 provides the user with the information that a grace period will expire in the future, and be followed by an enforcement period. By scheduling the updates for a convenient time within the grace period, the user is able to prevent an inconvenient update at the onset of the enforcement period. The user may therefore perform the update immediately (by selecting the first button) or schedule the update to be performed at a time of the user's choice (within the grace period) by pressing the second button. Additional information that will indicate, for example, when the update will be performed as a result of the end of the grace period and the beginning of the enforcement period, may be obtained by pressing the third button labeled "More Detail". Pressing the cancel button allows the user to return to work without resolving or changing the update installation time.

[0079] At block 1208, where the user selected the "Install Software Updates" button of Fig. 13 or the "Schedule Update" button of Fig. 14, a user interface 1500 is displayed, which allows the user to schedule the installation

of the updates. In the exemplary interface 1500, the user may click a check box 1502 to indicate agreement with the time selected via a pull-down menu 1504 to begin running the updates. Similarly, a check box 1506 may be used to indicate agreement with the time selected via the pull-down menu 1508 to perform a reboot. Alternatively, the pull-down menu 1508 may be used to indicate that the reboot should be performed immediately after update application.

[0080] At block 1210, a reminder interface, such as that seen in Fig. 14, may be displayed to the user at reboot or log-in, and periodically during the user's operation of the computer, to inform the user that required updates are pending and that the user may elect to schedule the updates at a convenient time.

[0081] At block 1212, the updates and reboot is performed at according to the user's elected schedule, or automatically, at the end of the grace period. At block 1214, in some instances, a reboot may not be performed. This may result because a document file (e.g. a word processing file) has been left open and unsaved. Alternatively, a reboot may not be performed where the user elected to postpone the reboot in response to an opportunity presented by a user interface. Where no reboot was performed, a repetitive user interface may be used to annoy the user until a reboot is performed. For example, Fig. 16 shows an exemplary annoyance mode interface 1600 which may be configured to repeatedly pop up at intervals to request that the user reboot. Accordingly, at block 1216 the user performs the reboot, thereby bringing the updates into service and ending the annoyance mode.

[0082] Fig. 17 is a flow diagram describing an exemplary method 1700 by which updates may be applied to a large number of systems in an orderly manner. For example, hundreds or thousands of servers may be configured to

host websites and perform commerce over the Internet. Organizational profitability may require that each server be up and running a set percentage of the time. Accordingly, the method 1700 results in a means by which updates may be applied to each server within a large group of servers within a tightly controlled period of time.

[0083] At block 1702, each server within a group of servers to be updated is associated into a subgroup. The subgroups are sized and configured to include servers in a manner which allows the simultaneous updating of servers in each subgroup without disrupting the work flow of the entire group. The time (e.g. 2am of the first Sunday of each month) when each subgroup may receive updates is assigned.

[0084] At block 1704, an anticipated elapsed time required for application of each update to be performed is calculated or measured. Where possible, the anticipated time may be adjusted according to expectations for each server, e.g. longer failsafe timeout period for slower servers. Using the anticipated time for application of each update, failsafe timeout periods are set for each update. For example, where an update is measured to take 5 minutes to install on a test machine, the failsafe timeout period may be set for 10 minutes. Accordingly, where the installation of that update reaches or exceeds 10 minutes, it will be assumed that the installation has failed, and the installation will be terminated with the appropriate success code. As a result, excessive time is not spent on the installation of any update.

[0085] At block 1706, the specific time period (i.e. the "change window") during which each subgroup of servers may be taken down for update installation in a given time period may be determined. In an exemplary application, each subgroup may be allocated a specific one-hour time slot of down time each month during which updates may be performed.

[0086] At block 1708, during the change window, i.e. the exact period of time having a specific starting and finishing time scheduled for updates, as many updates as possible are applied to the systems within the subgroup. During application of each update, the failsafe time is monitored, and where the failsafe time is exceeded, installation of the update is suspended and an appropriate success code is returned.

[0087] At block 1710, when time remaining within the change window is less than the time required for application of any remaining update and for rebooting, the installation of updates is suspended. The time required for application of each update may be based on the failsafe time. Accordingly, the system will be returned to service without exceeding the parameters of the change window.

[0088] At block 1712, any updates which failed to install during the change window (because, for example, the change window did not provide enough time to perform their install) are identified for potential installation in the next change window.

[0089] At block 1714, the server is rebooted and brought back on line.

[0090] Fig. 18 is a flow diagram describing an exemplary method 1800 by which the efficiency of update installation may be improved by configuring a unit of content—in one example configured within an SMS package—with a large number of updates.

[0091] At block 1802, a large number of updates are grouped into a single package. The package may be configured as an SMS package, and the aggregated updates may be considered to be a unit of content.

[0092] At block 1804, the package is configured with content obtained from a minimal number of trusted update authorities. In a preferred example, the content is received from a single update authority. By minimizing the

number of content providers, the cost of authenticating each provider and each update provided is minimized.

[0093] At block 1806, the package is configured to support differential enforcement as applied to different client computers. In one example, an XML document which is included with the updates within the package may provide different rules of enforcement for different client computers. The XML document may recite rules using Boolean operators or other means which result in application of the correct update(s) to each client. Accordingly, clients requiring updates within 24 hours may be treated differently than clients requiring updates within 30 days.

[0094] At block 1808, in one example, the package is configured for SMS consumption. Accordingly, at block 1810, SMS distributes the package to a plurality of clients, where the clients are associated with multiple service levels. For example, a more exacting service level may require a greater number of updates which are applied more rapidly after detection; a lower service level may require fewer updates and may provide additional time before they must be applied.

[0095] At block 1812, the updates within the package are applied to different client computers in a different manner, according to their service level, the time and duration of their change window, and other factors.

[0096] Fig. 19 is a flow diagram describing an exemplary method 1900 by which un-trusted updates may be tested and evaluated in an efficient manner.

[0097] At block 1902, a partition is formed in a package, wherein the package typically contains a large number of updates. At block 1904, an exemplary partition separates trusted updates from un-trusted updates. An update may be un-trusted, for example, due to lack of testing. Referring briefly

to Fig. 20, the update package 2000 may include trusted updates 2002—2004 and un-trusted updates 2006—2008 separated by a partition 2010. At block 1906, an exemplary partition is expressed using an XML file. The XML file is configured to inform different clients of the updates suitable for their consumption. In particular, the XML file may direct test systems to try un-trusted updates, and may direct production systems to avoid un-trusted updates.

[0098] At block 1908, an initial level of trust attributed to each update within the package is based on the performance of that update within a test environment. Following success of an un-trusted update in an appropriate number of systems over an appropriate period of time, the un-trusted update becomes a trusted update. At block 1910, un-trusted updates are merged with trusted updates after approval within the test environment. The merging of an un-trusted update with the trusted updates may be accomplished by editing the XML file which partitions the package.

[0099] Fig. 21 is a flow diagram describing an exemplary method 2100 by which a template may be constructed from a reference computer. The template reflects information on the update level of the reference computer, and facilitates update detection and deployment.

[00100] At block 2102, a file, such as a XML file, is opened to record information including updates approved for a reference computer. A reference computer is a computer having a disk with an image that is considered to be a standard within an organization, i.e. the disk contains a standard software configuration utilized by a corporation or other entity. The use of a reference computer simplifies management of systems within an organization by reducing the number of approved configurations allowed. For example, in the event of a problem, the disk may be reformatted with the standard image and the system restarted. At block 2104, in a template mode, a

template-making module scans a reference system and generates an authorization list. The authorization list includes all of the updates applied to the standard image on the reference system. At block 2106, the authorization list is incorporated into the template, which is written to the XML file. At block 2108, the template can be consumed and deployed as a mirror of the desired state. At block 2110, the deployed template is used to reduce the time elapsed from update detection to update deployment. Since the template identifies a subset of the updates needed from a large file such as update package 2000 in Fig. 20, the template can reduce the complexity of determining which updates are needed. Accordingly, at block 2112, a client computer is updated according to the template. In the updating process, updates (i.e. "patches") are selected in a more efficient manner by referring to the template, which results in client updating in a manner which is consistent with the organization's reference computer.

[00101] Although the disclosure has been described in language specific to structural features and/or methodological steps, it is to be understood that the appended claims are not limited to the specific features or steps described. Rather, the specific features and steps are exemplary forms of implementing this disclosure. For example, actions described in blocks of the flow diagrams may be performed in parallel with actions described in other blocks, the actions may occur in an alternate order, or may be distributed in a manner which associates actions with more than one other block. Moreover, the elements of the methods disclosed may be performed by any desired means, such as by the execution of processor-readable instructions defined on a processor-readable media, such as a disk, a ROM or other memory device, or by operation of an ASIC (application specific integrated circuit) or other hardware device.